# CS 553: Programming Assignment 1

## BENCHMARKING Design Document

**Viral Bhojani**
**Swapnil Dharawat**

**Introduction:**
Document consist of performance evaluation of 4 different benchmarks which are

1. CPU Benchmarking
2. DISK Benchmarking
3. MEMORY Benchmarking
4. NETWORK Benchmarking

**Consideration:**
Plots are generated using gnuplot. All the experiments were run on OpenStack KVM instances on Chameleon Testbed.
Specifications for the KVM we ran the benchmarks on:

| Specification of KVM | |
|---|---|
| **Flavor** | m1.medium |
| **Flavor ID** | 3 |
| **RAM** | 4GB |
| **VCPUs** | 2 VCPU |
| **Disk** | 40GB |

# Performance Evaluation

## 1. CPU Benchmarking

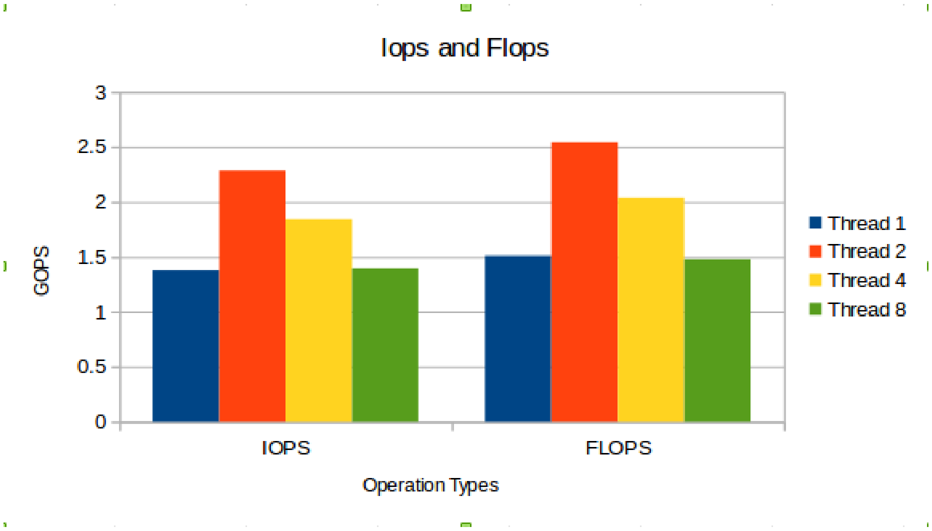
1.1 Decision:

- It is written in C programming language to generate GIOPS and GFLOPS by iterating through 40 operations for each thread (i.e. 1, 2, 4, 8) passed in array and choice for integer and float is also passed in array.
- Strong scaling is achieved as follows. For each number of threads all the 40 operations are executed. So, when 1 thread is passed 40 operations are executed by in thread and when 2 threads are passed, 20 operations are passed in thread 1 and 20 for thread 2.

1.2 Report:

| Type | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **GIOPS** | 1.376806 | 2.284295 | 1.843999 | 1.394237 |
| **GFLOPS** | 1.50988 | 2.543246 | 2.03924 | 1.479079 |

Graphical Representation:



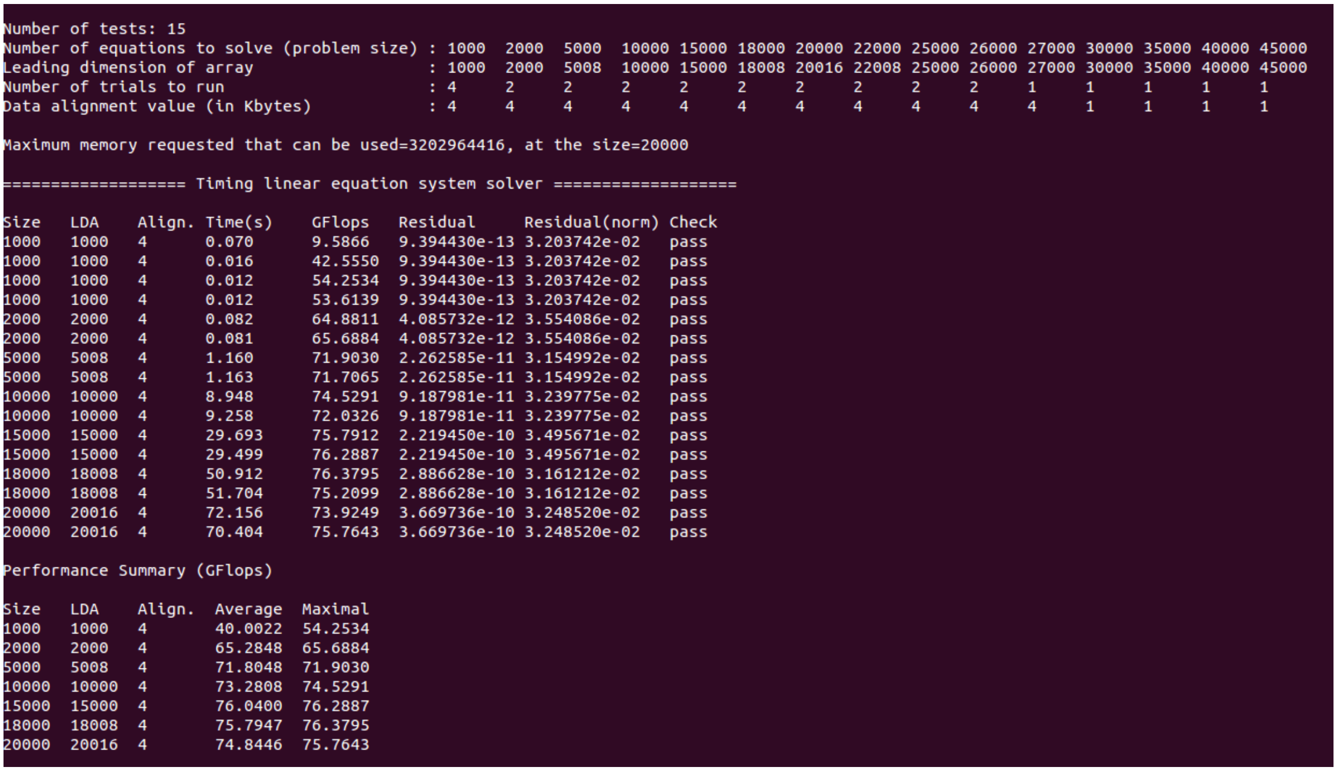
**Theoretical Performance:**

**CPU speed(GHz) * (Number of CPU cores) * IPC**
= 3.049 GHz * 2 * 2
= 12.196

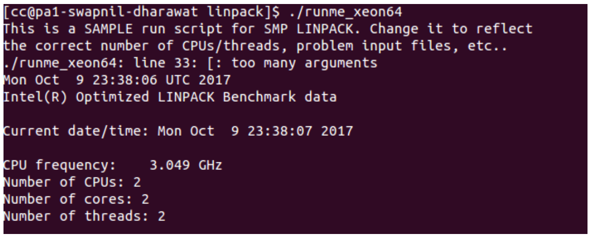
Compared to theoretical performance, achieved is around 20% FLOPS

In Ideal scenario, CPU utilization should be 100%. In our case it was nearly 20% of CPU utilization and hence the difference

**LINPACK benchmarking performance:**



```
Number of tests: 15
Number of equations to solve (problem size) : 1000  2000  5000  10000 15000 18000 20000 22000 25000 26000 27000 30000 35000 40000 45000
Leading dimension of array                  : 1000  2000  5008  10000 15000 18008 20016 22008 25000 26000 27000 30000 35000 40000 45000
Number of trials to run                     : 4     2     2     2     2     2     2     2     2     2     1     1     1     1     1
Data alignment value (in Kbytes)            : 4     4     4     4     4     4     4     4     4     4     4     1     1     1     1

Maximum memory requested that can be used=3202964416, at the size=20000

=================== Timing linear equation system solver ===================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
1000   1000   4      0.070      9.5866   9.394430e-13 3.203742e-02   pass
1000   1000   4      0.016      42.5550  9.394430e-13 3.203742e-02   pass
1000   1000   4      0.012      54.2534  9.394430e-13 3.203742e-02   pass
1000   1000   4      0.012      53.6139  9.394430e-13 3.203742e-02   pass
2000   2000   4      0.082      64.8811  4.085732e-12 3.554086e-02   pass
2000   2000   4      0.081      65.6884  4.085732e-12 3.554086e-02   pass
5000   5008   4      1.160      71.9030  2.262585e-11 3.154992e-02   pass
5000   5008   4      1.163      71.7065  2.262585e-11 3.154992e-02   pass
10000  10000  4      8.948      74.5291  9.187981e-11 3.239775e-02   pass
10000  10000  4      9.258      72.0326  9.187981e-11 3.239775e-02   pass
15000  15000  4      29.693     75.7912  2.219450e-10 3.495671e-02   pass
15000  15000  4      29.499     76.2887  2.219450e-10 3.495671e-02   pass
18000  18008  4      50.912     76.3795  2.886628e-10 3.161212e-02   pass
18000  18008  4      51.704     75.2099  2.886628e-10 3.161212e-02   pass
20000  20016  4      72.156     73.9249  3.669736e-10 3.248520e-02   pass
20000  20016  4      70.404     75.7643  3.669736e-10 3.248520e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
1000   1000   4       40.0022  54.2534
2000   2000   4       65.2848  65.6884
5000   5008   4       71.8048  71.9030
10000  10000  4       73.2808  74.5291
15000  15000  4       76.0400  76.2887
18000  18008  4       75.7947  76.3795
20000  20016  4       74.8446  75.7643
```



```
[cc@pa1-swapnil-dharawat linpack]$ ./runme_xeon64
This is a SAMPLE run script for SMP LINPACK. Change it to reflect
the correct number of CPUs/threads, problem input files, etc..
./runme_xeon64: line 33: [: too many arguments
Mon Oct  9 23:38:06 UTC 2017
Intel(R) Optimized LINPACK Benchmark data

Current date/time: Mon Oct  9 23:38:07 2017

CPU frequency:    3.049 GHz
Number of CPUs: 2
Number of cores: 2
Number of threads: 2
```

## 2. Memory Benchmarking:

2.1 Decisions:

- It is Written in C programming language using multithreading concept for different block sizes (8B, 8KB, 8MB, 80MB) on constant memory size i.e. 1.28 GB and Throughtput and latency is calculated in three operations i.e. sequential write(memset), random write(memset) and read+write(memcpy).
- Strong Scaling is achieved by diving memory size by block size and number of threads.

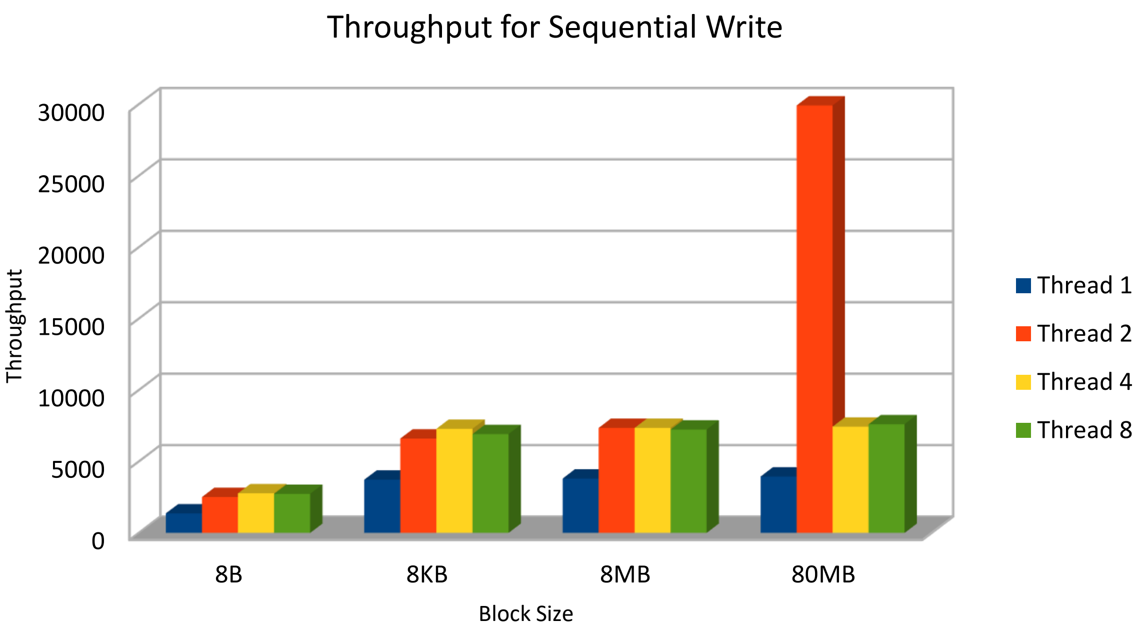
Report:

Throughput for Sequential Write, Sequential Read write and Random Write operations for (8B, 8KB, 8MB, 80MB) blocks and Threads 1, 2, 4, 8

Sequential Write

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **8B** | 1365.07732 | 2509.96242 | 2773.42371 | 2728.42096 |
| **8KB** | 3726.86497 | 6602.55517 | 7285.05588 | 6914.57604 |
| **8MB** | 3805.47605 | 7350.8756 | 7348.60581 | 7231.31701 |
| **80MB** | 3943.64973 | 29919.1722 | 7434.98013 | 7606.31873 |

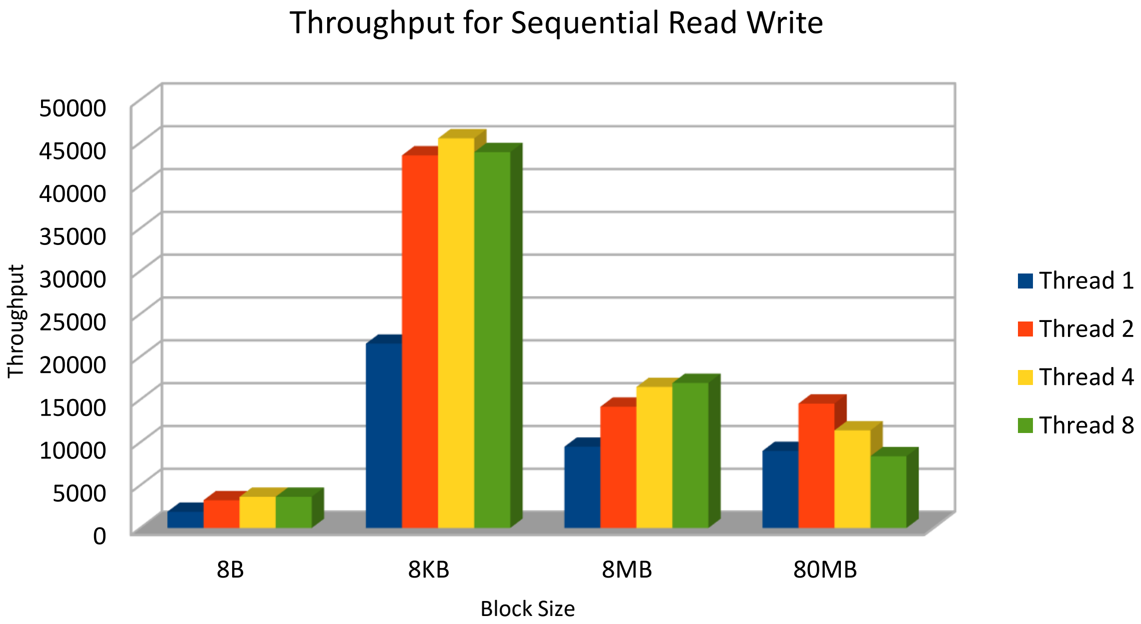
Graphical Representation



Sequential Read Write

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **8B** | 1893.37628 | 3235.33076 | 3634.64825 | 3639.76665 |
| **8KB** | 21520.0622 | 43506.7635 | 45495.2758 | 43876.5654 |
| **8MB** | 9486.56303 | 14153.8984 | 16463.649 | 16929.9931 |
| **80MB** | 8992.31248 | 14524.8224 | 11393.1001 | 8376.22311 |

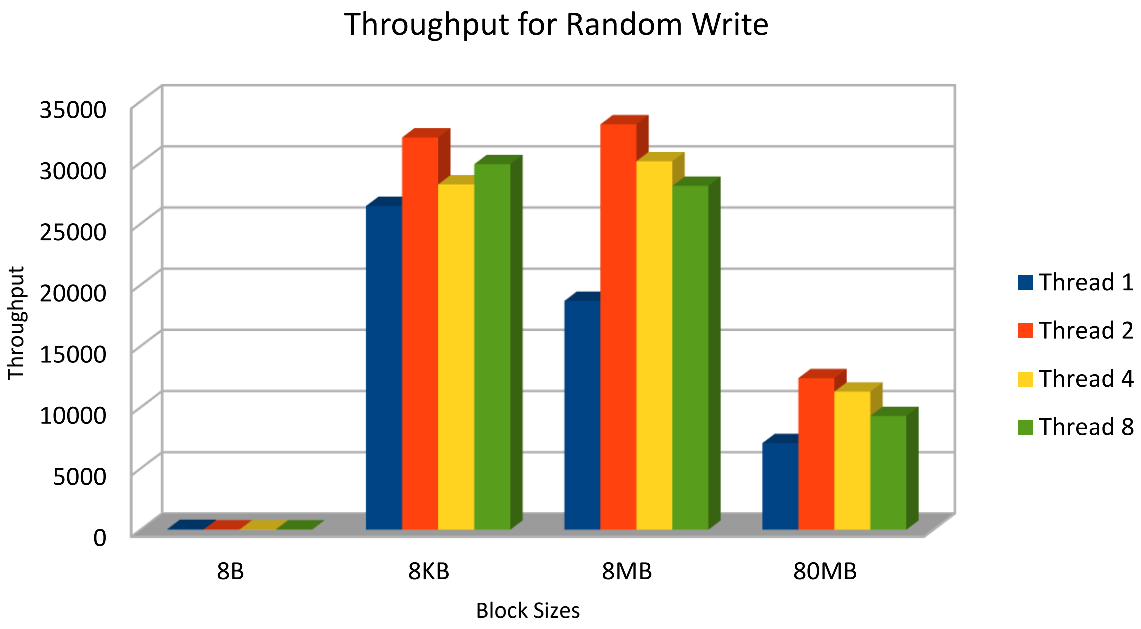
Graphical Representation:



Random Write:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **8B** | 62.020897 | 27.470192 | 25.963327 | 24.510463 |
| **8KB** | 26459.9538 | 32061.9019 | 28220.8267 | 29876.7344 |
| **8MB** | 18702.6761 | 33133.3888 | 30117.0053 | 28102.8209 |
| **80MB** | 7077.28629 | 12384.3423 | 11285.0982 | 9294.76851 |

Graphical Representation:

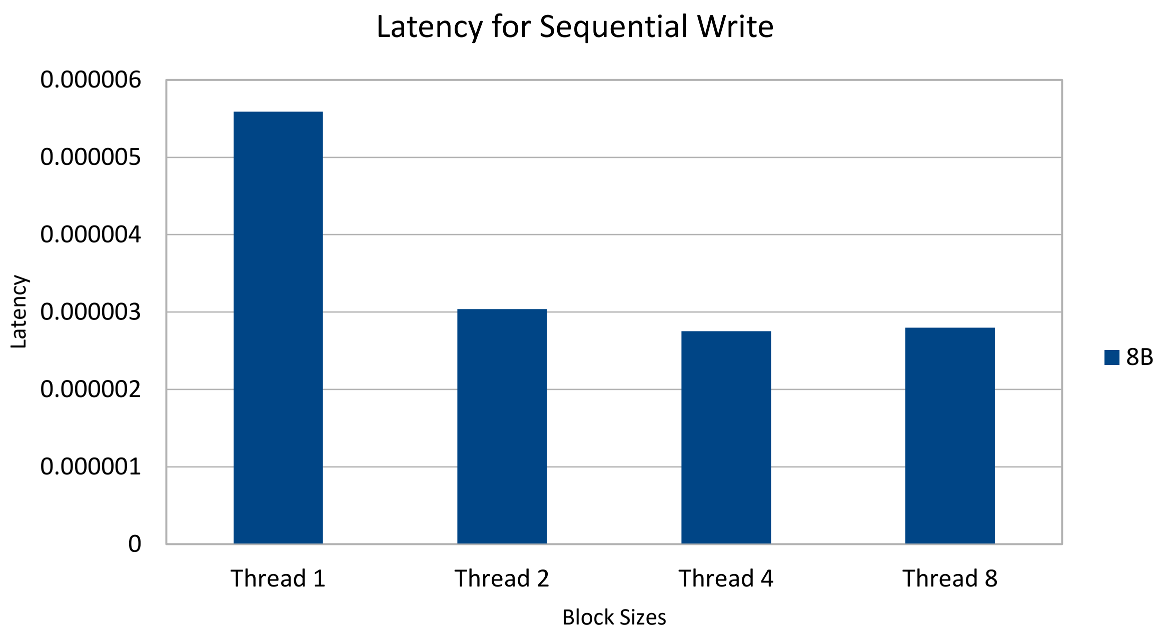


Latency for Sequential Write, Sequential Read write and Random Write operations for (8B) blocks and Threads 1, 2, 4, 8

Sequential Write:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **8B** | 5.589E-06 | 3.0396E-06 | 2.7509E-06 | 2.7963E-06 |

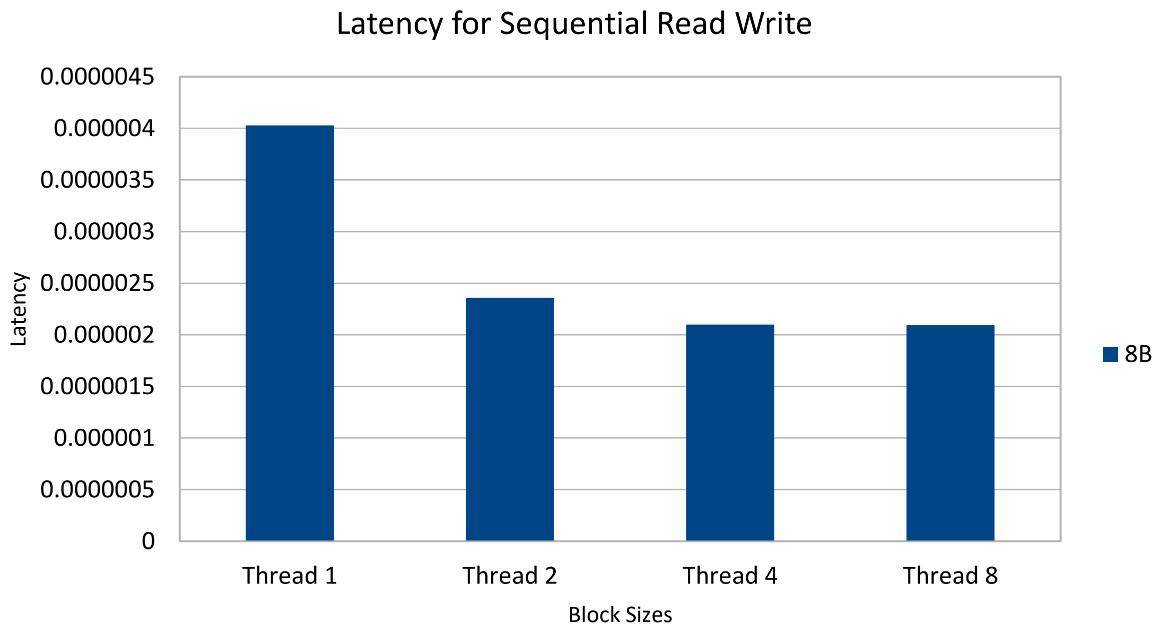
Graphical Representation:



Sequential Read Write:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **8B** | 4.0295E-06 | 2.3582E-06 | 2.0991E-06 | 2.0961E-06 |

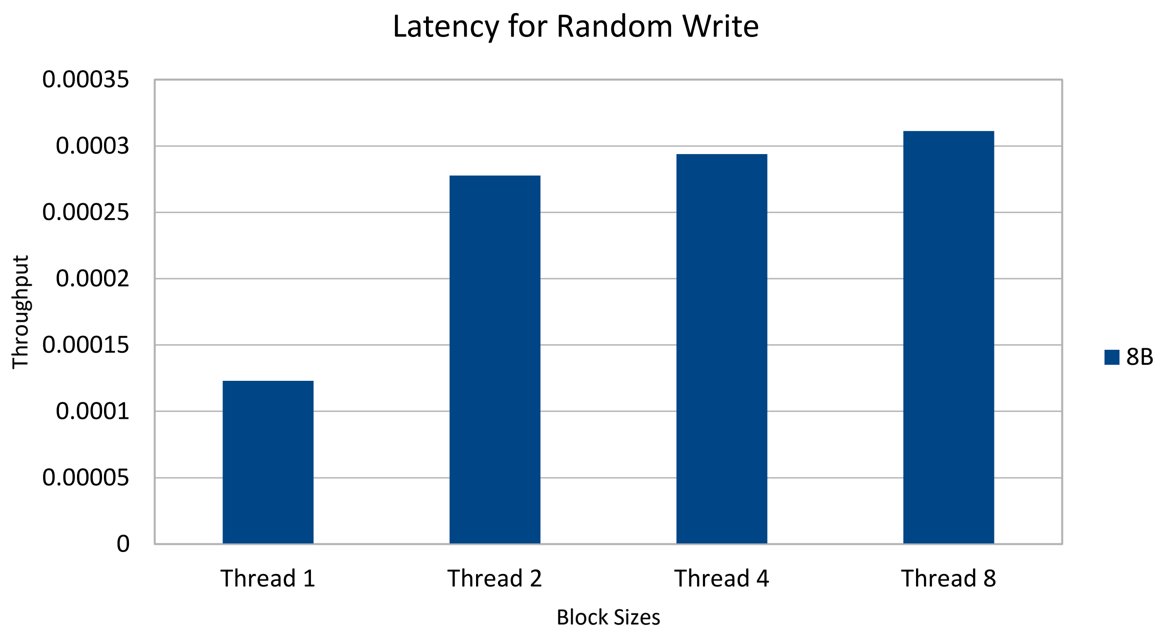
Graphical Representation:



Random Write:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **8B** | 0.00012301 | 0.00027773 | 0.00029385 | 0.00031127 |

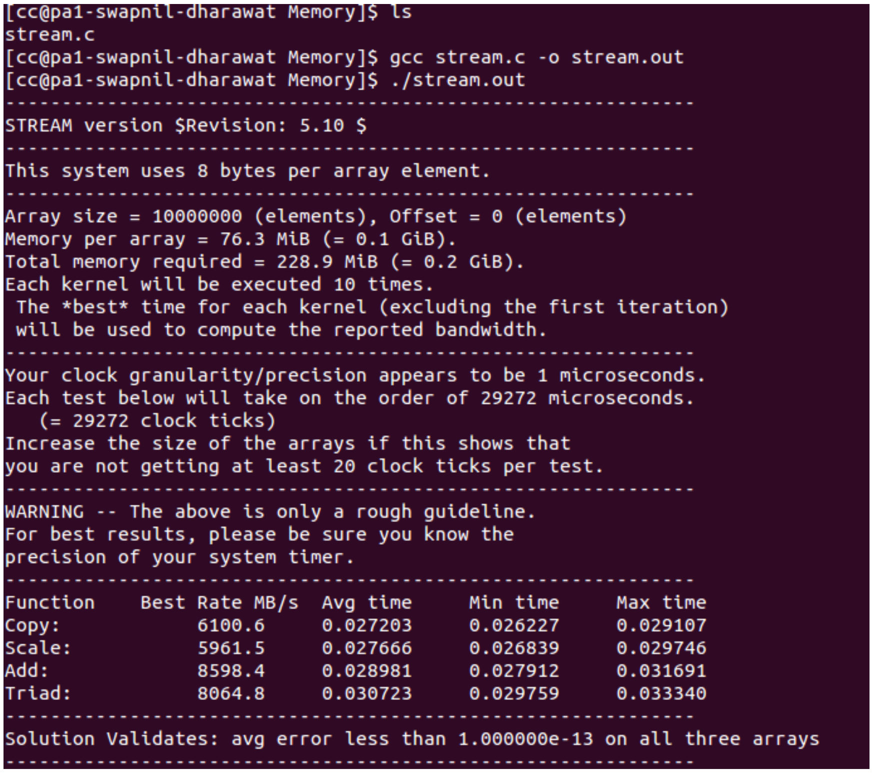
Graphical Representation:



**Theoretical Performance:**

**Clock Frequency * Number of data transfer per clock * Memory bus interface width * number of interfaces**

**Stream Test**



```
[cc@pa1-swapnil-dharawat Memory]$ ls
stream.c
[cc@pa1-swapnil-dharawat Memory]$ gcc stream.c -o stream.out
[cc@pa1-swapnil-dharawat Memory]$ ./stream.out
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 10000000 (elements), Offset = 0 (elements)
Memory per array = 76.3 MiB (= 0.1 GiB).
Total memory required = 228.9 MiB (= 0.2 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 29272 microseconds.
   (= 29272 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:            6100.6     0.027203     0.026227     0.029107
Scale:           5961.5     0.027666     0.026839     0.029746
Add:             8598.4     0.028981     0.027912     0.031691
Triad:           8064.8     0.030723     0.029759     0.033340
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```

We ran a Stream benchmark and compared all the results obtained. The thing is Stream doesn't give much flexibility to run all experiments with different parameters.
The result when run on Chameleon Testbed is divided into 4 parts having total of 28.05 GB/s.

3. Disk Benchmarking:

3.1 Decision:

- It is Written in C programming language using multithreading concept for different block sizes (8B, 8KB, 8MB, 80MB) on constant file size i.e. 10 GB which read and write into the file and Throughtput and latency is calculated in three operations i.e. sequential read(memset), random read(memset) and read+write(memcpy).
- Strong Scaling is achieved by diving memory size by block size and number of threads.

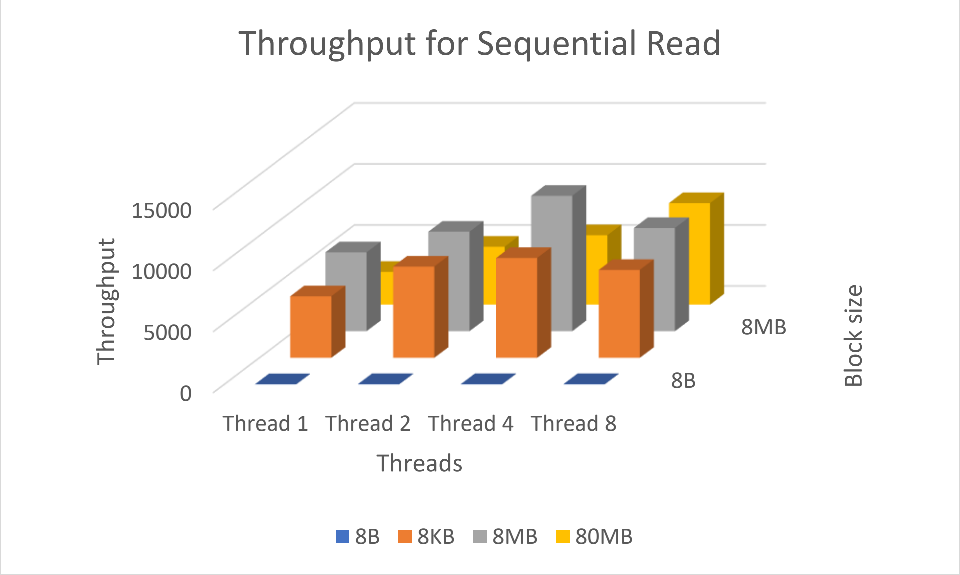
3.2 Report

Throughput for Sequential Read, Sequential Read Write and Random Read operations for (8B, 8KB, 8MB, 80MB) blocks and Threads 1, 2, 4, 8

Sequential Read:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| 8B | 16.213795 | 25.795656 | 27.089802 | 28.316206 |
| 8KB | 5042.983555 | 7476.984906 | 8178.652439 | 7199.829004 |
| 8MB | 6461.415757 | 8155.30764 | 11098.58666 | 8456.771363 |
| 80MB | 2681.363893 | 4743.446262 | 5686.51319 | 8317.305193 |

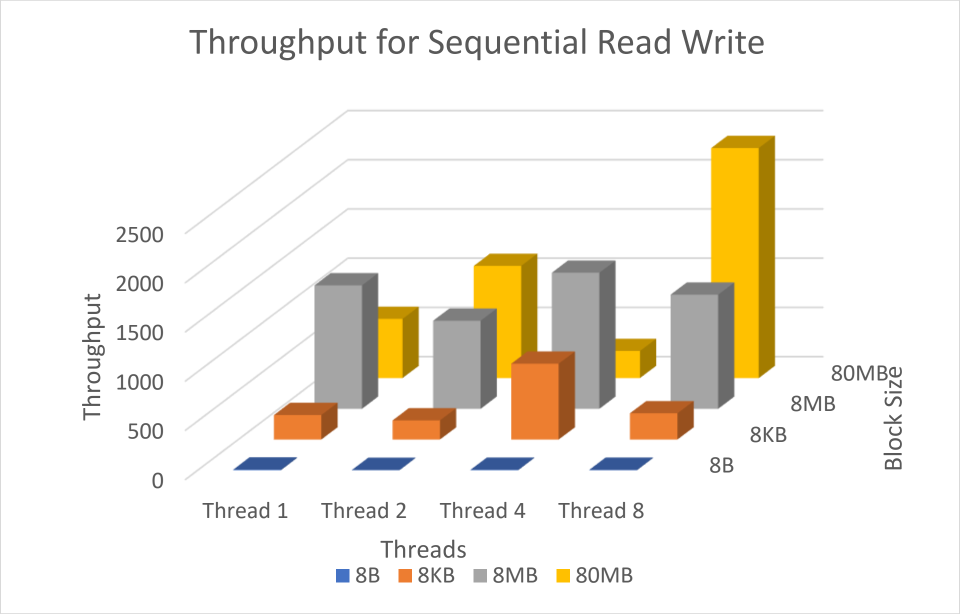
Graphical Representation:



Sequential Read Write:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| 8B | 6.890572 | 3.067626 | 4.719271 | 4.40189 |
| 8KB | 248.669376 | 194.182122 | 770.381248 | 266.575254 |
| 8MB | 1253.830059 | 894.463758 | 1382.343158 | 1159.407688 |
| 80MB | 601.537398 | 1139.604183 | 275.645651 | 2337.067163 |

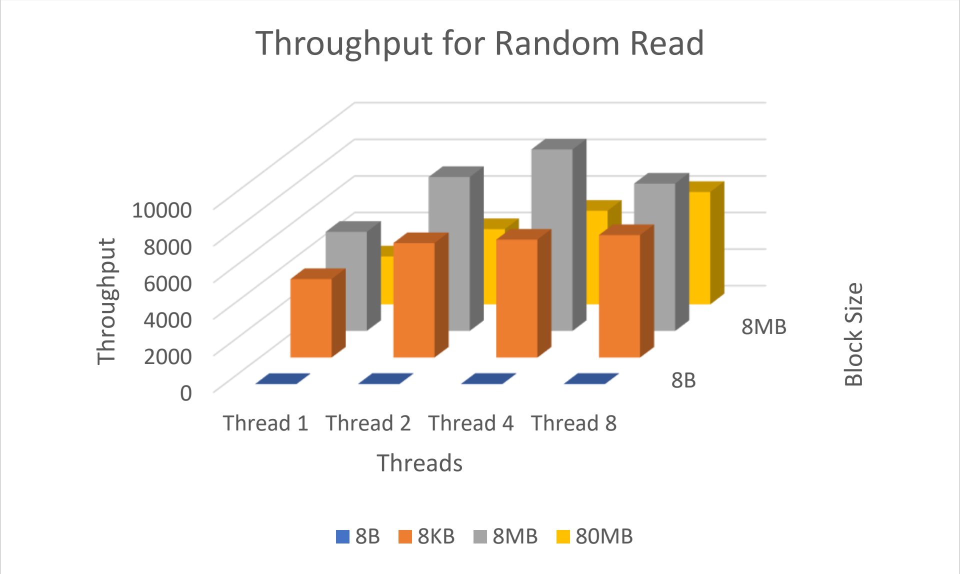
Graphical Representation:



Random Read:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| 8B | 12.239864 | 15.55315 | 17.524179 | 16.642717 |
| 8KB | 4289.414863 | 6254.642117 | 6442.553063 | 6691.026184 |
| 8MB | 5415.170091 | 8408.329501 | 9916.254135 | 8053.404136 |
| 80MB | 2620.40559 | 4112.793358 | 5113.740781 | 6138.91207 |

Graphical Representation:

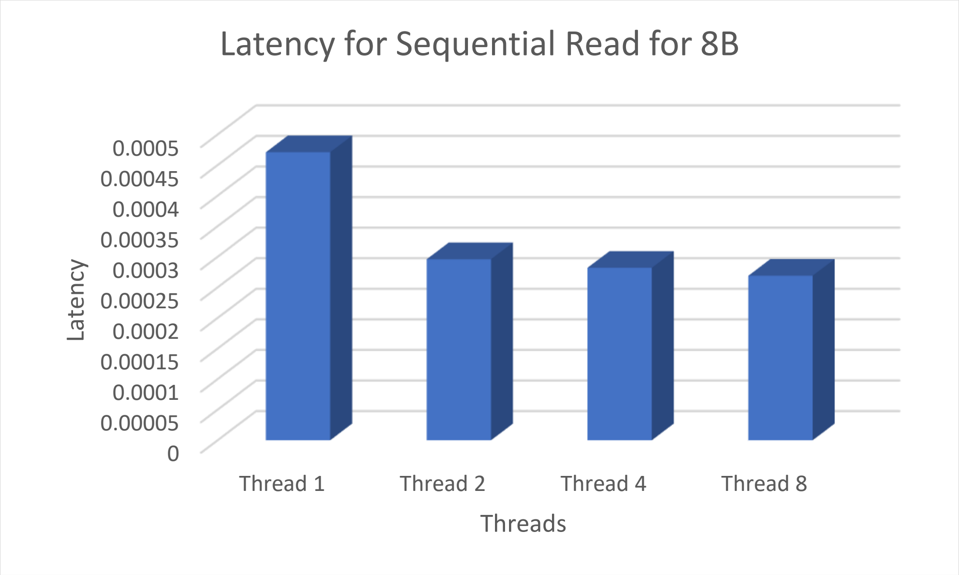


Latency for Sequential Read, Sequential Read Write and Random Read operations for (8B) blocks and Threads 1, 2, 4, 8

Sequential Read:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **8B** | 0.000471 | 0.000296 | 0.000282 | 0.000269 |

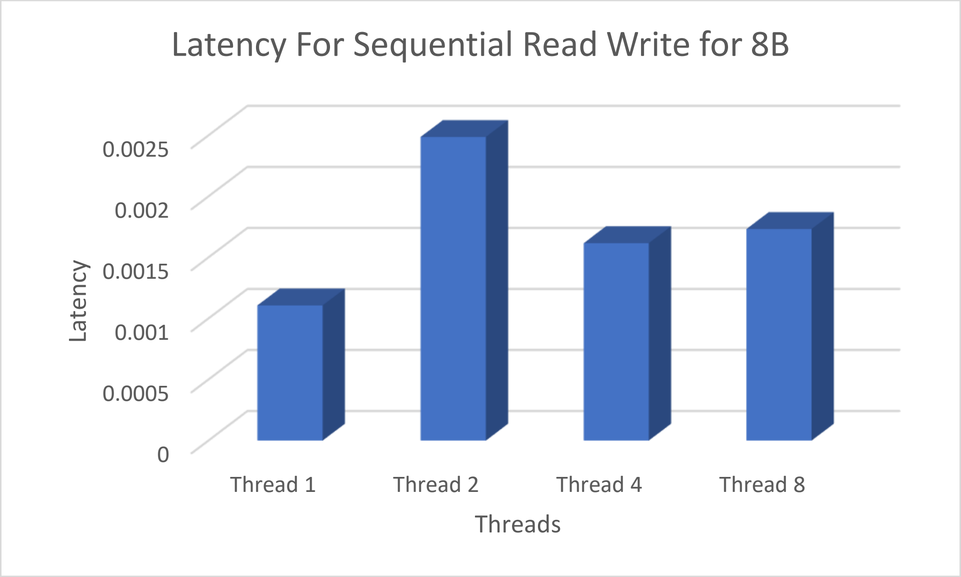
Graphical Representation:



Sequential Read Write:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **8B** | 0.001107 | 0.002487 | 0.001617 | 0.001733 |

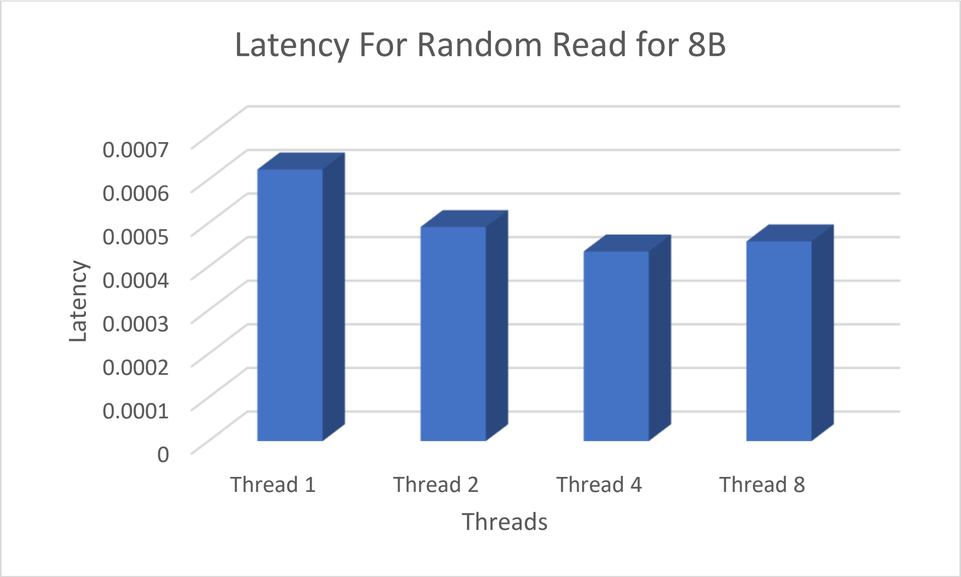
Graphical Representation:



Random Read:

| Block Size | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| **8B** | 0.000623 | 0.000491 | 0.000435 | 0.000458 |

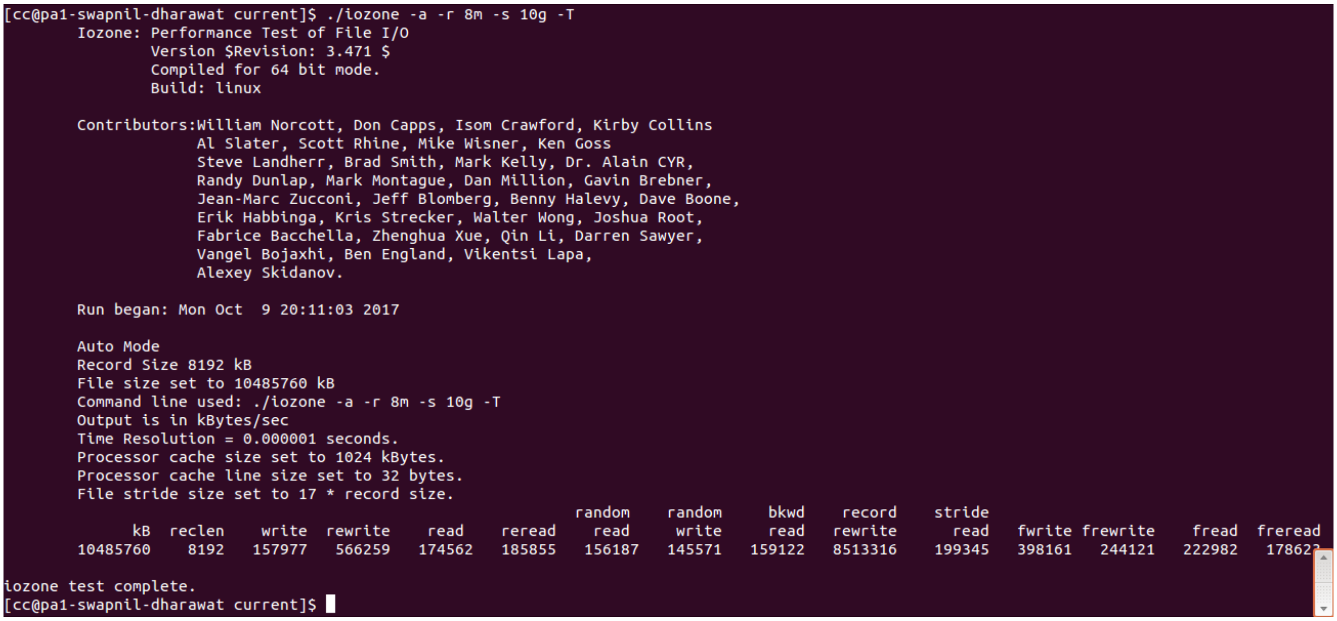
Graphical Representation:



IoZone Benchmarking;

The benchmarking is performed on two sets

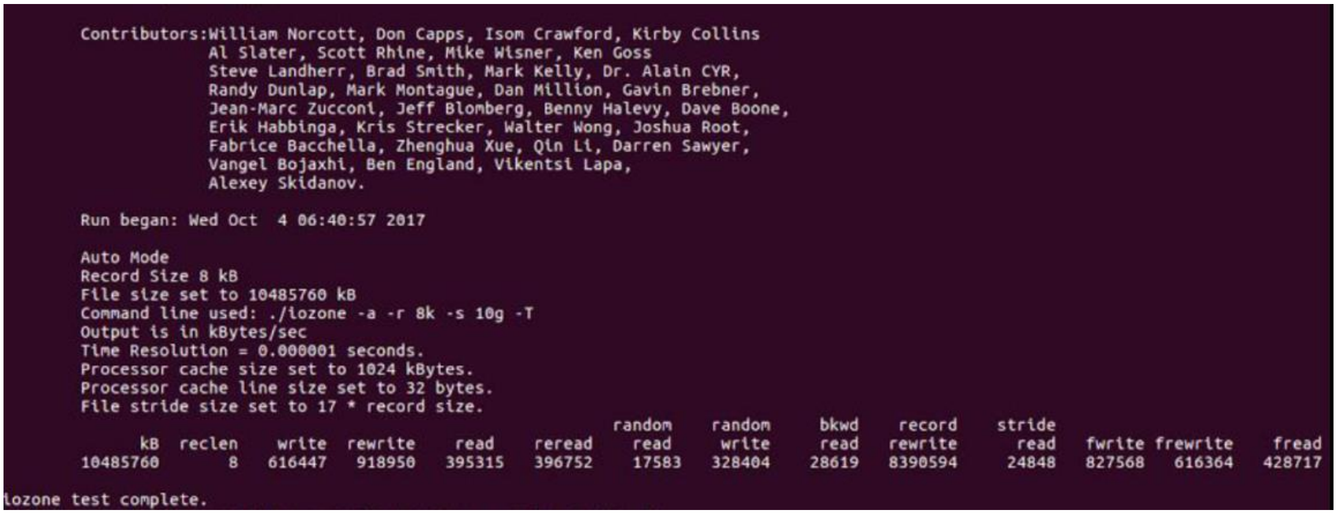
i. 8 MB block size and 10GB File size



```
[cc@pa1-swapnil-dharawat current]$ ./iozone -a -r 8m -s 10g -T
        Iozone: Performance Test of File I/O
                Version $Revision: 3.471 $
                Compiled for 64 bit mode.
                Build: linux

        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million, Gavin Brebner,
                     Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
                     Erik Habbinga, Kris Strecker, Walter Wong, Joshua Root,
                     Fabrice Bacchella, Zhenghua Xue, Qin Li, Darren Sawyer,
                     Vangel Bojaxhi, Ben England, Vikentsi Lapa,
                     Alexey Skidanov.

        Run began: Mon Oct  9 20:11:03 2017

        Auto Mode
        Record Size 8192 kB
        File size set to 10485760 kB
        Command line used: ./iozone -a -r 8m -s 10g -T
        Output is in kBytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 kBytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
                                                              random    random     bkwd    record    stride
              kB  reclen    write  rewrite    read    reread     read    write     read   rewrite      read   fwrite frewrite    fread  freread
        10485760    8192   157977   566259  174562    185855   156187   145571   159122   8513316    199345   398161   244121   222982   17862

iozone test complete.
[cc@pa1-swapnil-dharawat current]$ 
```

The performance are as follows:
Sequential read operation is 174562 MBPS, Read+ write 566259 MBPS and Random read is 156187 MBPS
Our performance is 36% of ideal scenario when compared on Random read

ii. 8KB block and 10 GB file Size



```
        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million, Gavin Brebner,
                     Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
                     Erik Habbinga, Kris Strecker, Walter Wong, Joshua Root,
                     Fabrice Bacchella, Zhenghua Xue, Qin Li, Darren Sawyer,
                     Vangel Bojaxhi, Ben England, Vikentsi Lapa,
                     Alexey Skidanov.

        Run began: Wed Oct  4 06:40:57 2017

        Auto Mode
        Record Size 8 kB
        File size set to 10485760 kB
        Command line used: ./iozone -a -r 8k -s 10g -T
        Output is in kBytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 kBytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
                                                              random    random      bkwd    record     stride
              kB  reclen    write  rewrite     read    reread     read     write      read   rewrite       read   fwrite frewrite     fread
        10485760       8   616447   918950   395315   396752    17583    328404     28619   8390594      24848   827568   616364    428717

iozone test complete.
```

The performance are as follows:
Sequential read operation is 395313 MBPS, Read+ write 918950 MBPS and Random read is 17583 MBPS
Our performance is 34% of ideal scenario when compared on Random read.

4. Network Benchmarking:

4.1 Decisions

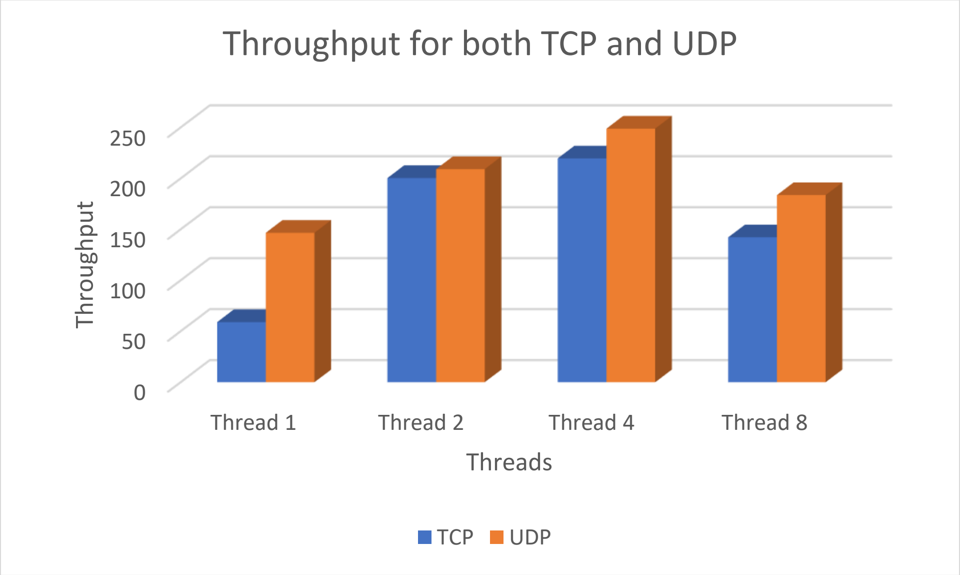
It is written in java to perform ping pong operation on both TCP and UDP using multithreading concept on both Client side and Server side to send data on the file upto 64KB from the client using port 5004 and server send response of the same file size.

4.2 Report:

Throughput value for both TCP and UDP for 1, 2, 4, 8 Threads.

| Connection Type | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| TCP | 58.99534481 | 200.4197528 | 219.6658307 | 142.2203617 |
| UDP | 146.6576426 | 209.121812 | 248.8857038 | 183.6398957 |

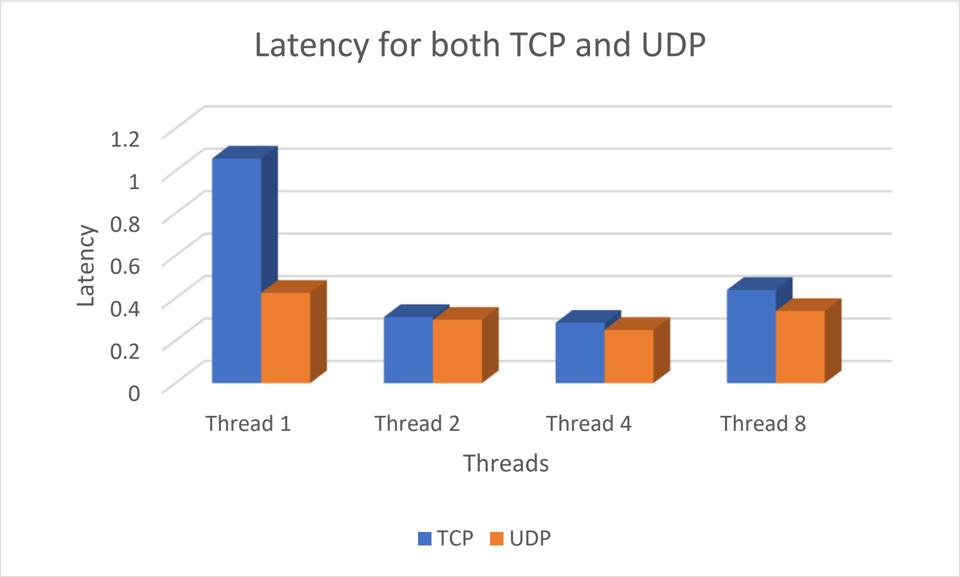
Graphical Representation:



Latency value for both TCP and UDP for 1, 2, 4, 8 Threads.

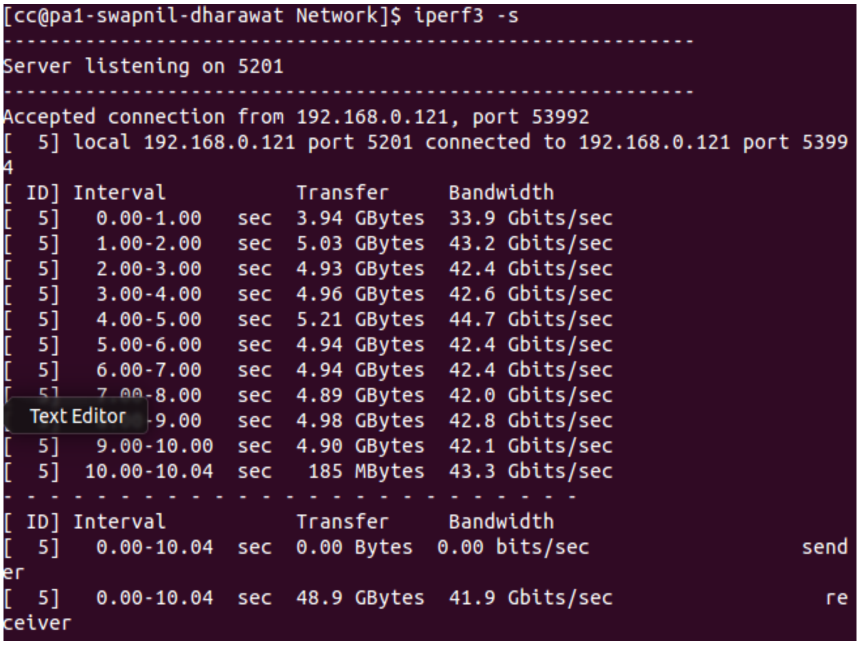
| Connection Type | Thread 1 | Thread 2 | Thread 4 | Thread 8 |
|---|---|---|---|---|
| TCP | 1.0589045 | 0.311698 | 0.2843885 | 0.439251 |
| UDP | 0.425961 | 0.2987275 | 0.25100049 | 0.340179 |

Graphical Representation:



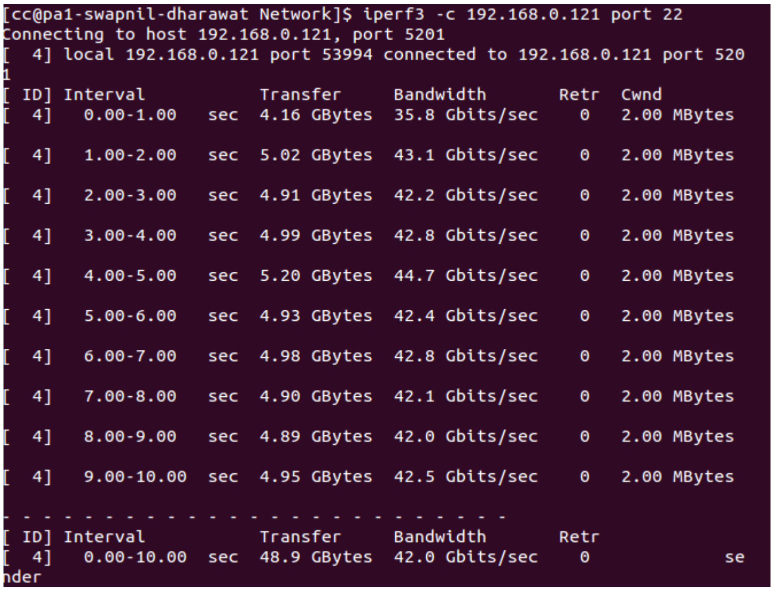
**Iperf Benchmarking:**

**Server side:**



```
[cc@pa1-swapnil-dharawat Network]$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.0.121, port 53992
[  5] local 192.168.0.121 port 5201 connected to 192.168.0.121 port 5399
4
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  3.94 GBytes  33.9 Gbits/sec
[  5]   1.00-2.00   sec  5.03 GBytes  43.2 Gbits/sec
[  5]   2.00-3.00   sec  4.93 GBytes  42.4 Gbits/sec
[  5]   3.00-4.00   sec  4.96 GBytes  42.6 Gbits/sec
[  5]   4.00-5.00   sec  5.21 GBytes  44.7 Gbits/sec
[  5]   5.00-6.00   sec  4.94 GBytes  42.4 Gbits/sec
[  5]   6.00-7.00   sec  4.94 GBytes  42.4 Gbits/sec
[  5]   7.00-8.00   sec  4.89 GBytes  42.0 Gbits/sec
  Text Editor   -9.00   sec  4.98 GBytes  42.8 Gbits/sec
[  5]   9.00-10.00  sec  4.90 GBytes  42.1 Gbits/sec
[  5]  10.00-10.04  sec   185 MBytes  43.3 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-10.04  sec  0.00 Bytes  0.00 bits/sec                  send
er
[  5]   0.00-10.04  sec  48.9 GBytes  41.9 Gbits/sec                  re
ceiver
```

**Client side:**



```
[cc@pa1-swapnil-dharawat Network]$ iperf3 -c 192.168.0.121 port 22
Connecting to host 192.168.0.121, port 5201
[  4] local 192.168.0.121 port 53994 connected to 192.168.0.121 port 520
1
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  4.16 GBytes  35.8 Gbits/sec    0   2.00 MBytes

[  4]   1.00-2.00   sec  5.02 GBytes  43.1 Gbits/sec    0   2.00 MBytes

[  4]   2.00-3.00   sec  4.91 GBytes  42.2 Gbits/sec    0   2.00 MBytes

[  4]   3.00-4.00   sec  4.99 GBytes  42.8 Gbits/sec    0   2.00 MBytes

[  4]   4.00-5.00   sec  5.20 GBytes  44.7 Gbits/sec    0   2.00 MBytes

[  4]   5.00-6.00   sec  4.93 GBytes  42.4 Gbits/sec    0   2.00 MBytes

[  4]   6.00-7.00   sec  4.98 GBytes  42.8 Gbits/sec    0   2.00 MBytes

[  4]   7.00-8.00   sec  4.90 GBytes  42.1 Gbits/sec    0   2.00 MBytes

[  4]   8.00-9.00   sec  4.89 GBytes  42.0 Gbits/sec    0   2.00 MBytes

[  4]   9.00-10.00  sec  4.95 GBytes  42.5 Gbits/sec    0   2.00 MBytes

- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  48.9 GBytes  42.0 Gbits/sec    0             se
nder
```